## **Platform Engineer - Case Study**

### **Objective**

The goal of this challenge is to evaluate your ability to design, implement, and deploy a solution that aligns with the kind of work you’ll do as a Platform Engineer on our team. This task will assess your skills in **Node.js**, **Kubernetes**, **MongoDB**, and **Kafka**, as well as your system design and problem-solving capabilities.

### **Scenario**

You are tasked with creating a minimal **Event-Driven Application** that processes data and exposes a simple API for querying the results. The project should demonstrate your ability to integrate **Kubernetes**, **Kafka**, **MongoDB**, and **Node.js**.

The systems in this scope comprises:

1. A Node.js app in the producer role which publishes every 3 seconds the payload included in task detail.
2. A Node.js app in the consumer role which saves incoming payload to the MongoDB.
3. A Node.js app in REST API role which has an endpoint to list events with filtering feature.
4. Custom Helm chart to deploy all software packages and required services.

### **Task #1 - Producer app**

Implement a Node.js app that publishes a payload filled with random values in the following schema. The app should be dockerized, publish every 3 seconds, and should read Kafka related configuration from environment variables. It should log published events in JSON format to the stdout.

Payload:

| {  "eventId": "string", // UUID V4 preferred  "eventType": "string", // e.g., "user\_signup", "order\_created"  "timestamp": "ISO8601 string",  "payload": "JSON object" // should at least have one attribute filled with random value  } |
| --- |

### **Task #2 - Consumer app**

Implement a Node.js app that consumes data from a given Kafka topic and stores it to MongoDB. The app should be dockerized, and should read Kafka and MongoDB related configuration from environment variables. It should log received events in JSON format to the stdout.

### **Task #3 - API app**

Implement a Node.js app that provides an endpoint for the following. The app should be dockerized, and should read MongoDB related configuration from environment variables. It should log served requests (path and headers would suffice, no need to log response body) in JSON format to the stdout.

* List endpoint returns all events that stored in MongoDB with following features
  + Filter events by eventType
  + Fetch events within a specific timestamp range
  + Pagination will be a plus

### **Task #3 - Infrastructure**

All applications must be dockerized, and can be deployable to Kubernetes via a custom Helm chart. Bitnami’s or other open sourced Helm charts can be used for MongoDB and Kafka.

### **Bonus Points**

* Metrics sidecar or endpoints that could be scraped by Prometheus will be a plus.
* Implement retries and dead-letter queues for the Kafka consumer.
* Docker Compose config for development teams to run all on local environments.
* Use Terraform to spin up a new Kubernetes cluster.

### **Deliverables**

1. A **GitHub repository** with:
   * All application source code.
   * Kubernetes manifests and Helm chart(s).
   * Documentation on how to run and test the application locally.
2. A short **document** that explains:
   * How you would ensure scalability, fault tolerance, and security.
   * Challenges faced and how you solved them.